MUIA

API

Versão 1.2

Histórico de Revisões

Versão	Data	Autor	Motivo
1.0	15/06/2015	Cleber Alcântara	Versão Inicial da API
1.1	22/06/2015	Cleber Alcântara	- Adicionando valores de erros - Corrigindo parâmetros - Revisando operações
1.2	23/06/2015	Cleber Alcântara	- Formatando corretamente o documento - Inserindo índice - Modificando parâmetros da classe ConnectionHeader - Atualizando imagem do diagrama de classes
1.3	24/06/2015	Cleber Alcântara	- Adicionando informação de obrigatoriedade de uma instância de MessageData para os tipos de MessageHeader existentes.

Índice

1 Definição

A API do MUIA (Middleware Unificado para Integração de Aplicações) é a porta de comunicação entre aplicações e o MUIA para permitir que essas aplicações e os MUIAS possam executar troca de mensagens entre si, viabilizando a integração dessas aplicações. Através dela, é possível especificar as ações e os dados utilizados em cada uma dessas ações através um conjunto de pacotes que se completam e formam as mensagens que são trocadas utilizando sockets.

2 Objetivo

Este documento tem por objetivo documentar os pacotes utilizados na comunicação com o MUIA e orientar essas aplicações a interagir com o MUIA.

3 Visão geral

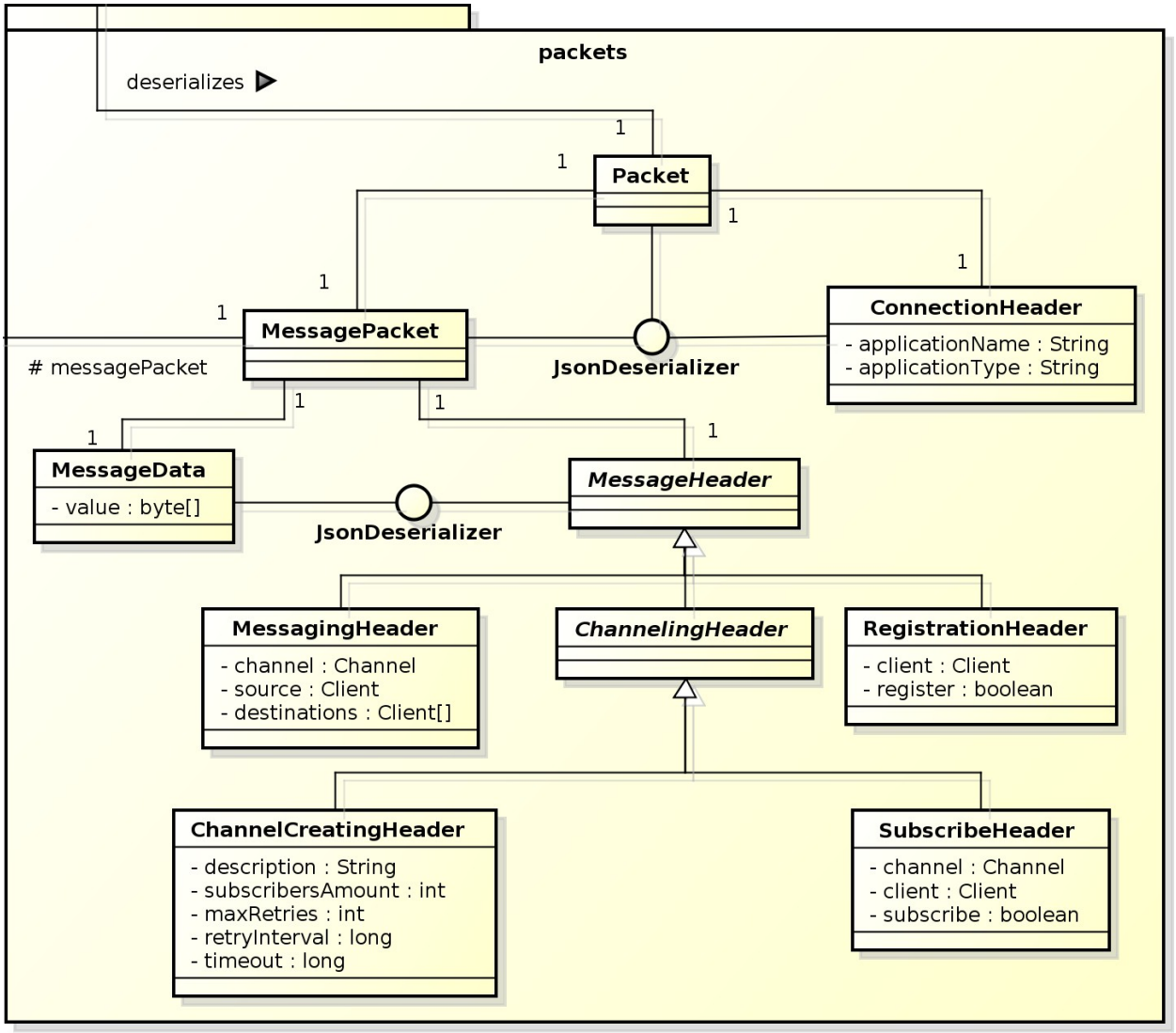

Figura 1: Hierarquia de classes de uma mensagem

Na Figura 1, é apresentada a hierarquia de classes que formam uma mensagem interpretável pelo MUIA. Essa imagem pode ser encontrada em melhor resolução neste link. Quando aplicações desejarem se comunicar com o MUIA, elas precisarão enviar mensagens que sigam essa hierarquia e estrutura serializada em JSON, com as operações e dados que elas desejarem. Detalhes sobre o formato JSON dessa hierarquia são apresentados em seções abaixo.

3.1 Classe Packet

3.1.1 Descrição

Esta é a classe raiz de uma mensagem que deve ser enviada para um MUIA. Ela deve conter os seguintes dados:

- **Valor da Mensagem**: Uma instância de ConnectionHeader, definida na seção 3.2;
- **Pacote de Mensagem**: Uma instância de MessagePacket, definida na seção 3.3.

3.1.2 JSON

O formato JSON dessa classe é conforme é apresentado abaixo:

```
{
    “connection-header” : <objeto ConnectionHeader serializado>,
    “message-packet” : <objeto MessagePacket serializado>
}
```

3.1.3 Erros

A desserialização dessa classe pode gerar os seguintes erros:

- MISSING_ARGUMENT: Caso ConnectionHeader ou o MessagePacket não seja informado;
- INVALID_VALUE: Caso ConnectionHeader ou MessagePacket desserializado apresente um valor inválido;
- INVALID_FORMAT: Caso o JSON informado para a classe apresente um formato inválido.

3.2 Classe ConnectionHeader

3.2.1 Descrição

Esta classe é responsável por identificar a aplicação que se conectou ao MUIA e está enviando a mensagem. Ela deve conter seguintes dados, através dos quais o MUIA será capaz de criar a instância da aplicação que se conecta a ele:

- **Tipo da Aplicação**: Uma string obrigatória que identifica se a aplicação que está se conectando é um MUIA (“`muia`”) ou Cliente (“`client`”);
- **Nome da Aplicação**: Uma string obrigatória que identifica o nome da aplicação que está se conectando ao MUIA;

3.2.2 JSON

O formato JSON dessa classe é conforme é apresentado abaixo:

```
{
     “app-type” : <Tipo da aplicação>,
     “app-name” : <Nome da aplicação>
}
```

3.2.3 Erros

A desserialização dessa classe pode gerar os seguintes erros:

- MISSING_ARGUMENT: Caso qualquer um dos dados que compõem a mensagem não seja informado;
- INVALID_VALUE: Caso `app-type` apresente valor inválido;
- INVALID_FORMAT: Caso o JSON informado para a classe apresente um formato inválido.

3.3 Classe MessagePacket

3.3.1 Descrição

Esta classe define o conteúdo da mensagem que será tratada pelo MUIA. Ela deve conter os seguintes dados:

- **Cabeçalho de Mensagem**: Uma instância de MessageHeader obrigatória, definida na seção 3.4;
- **Tipo de Cabeçalho de Mensagem**: Uma string obrigatória que identifica o tipo do cabeçalho de mensagem informado, que pode assumir os seguintes valores:
  - `registration`: Define um cabeçalho de adição ou remoção de registro de um Cliente no MUIA;
  - `channel-subscribing`: Define um cabeçalho de inscrição ou desinscrição de um Cliente em um canal;
  - `channel-creating`: Define um cabeçalho de criação de um canal;
  - `messaging`: Define um cabeçalho de envio de mensagens entre aplicações.
- **Dados da Mensagem**: Uma instância de MessageData obrigatória apenas quando o tipo do cabeçalho é `messaging`, definida na seção 3.10.

3.3.2 JSON

O formato JSON dessa classe é conforme é apresentado abaixo:

```
{
     “header-type” : <tipo de cabeçalho da mensagem>,
     “header-data” : <objeto MessageHeader serializado>,
     “message-data” : <objeto MessageData serializado>
}
```

3.3.3 Erros

A desserialização dessa classe pode gerar os seguintes erros:

- MISSING_ARGUMENT: Caso qualquer um dos dados que compõem a mensagem não seja informado;
- INVALID_VALUE: Caso qualquer um dos dados que compõem a mensagem não apresentem um valor válido;
- INVALID_FORMAT: Caso o JSON informado para a classe apresente um formato inválido.

3.4 Classe MessageHeader

3.4.1 Descrição

Como pode ser observado na Figura 1, esta classe é abstrata. Isso se deve ao fato de que ela apenas define uma tipo comum a todos os cabeçalhos de mensagens aceitos pelo MUIA. Sendo assim, essa classe não apresenta uma serialização específica, pois essa serialização parte das suas subclasses concretas.

3.5 Classe MessagingHeader

3.5.1 Descrição

Esta classe define o cabeçalho de uma mensagem que tem por objetivo enviar uma mensagem a uma ou mais aplicações. Ela deve conter os seguintes dados

- **Identificação do Canal da Mensagem**: Uma string obrigatória que identifica canal onde a aplicação será publicada;
- **Identificação da Aplicação de Origem**: Uma string obrigatória que identifica a aplicação remetente da mensagem;
- **Identificações das Aplicações de Destino**: Um lista de strings que identificam as aplicações destinatárias da mensagem. Este dado pode ser:
    - Vazio ou nulo, quando deseja-se enviar a mensagem para todas as aplicações daquele canal;
    - Um único valor, quando deseja-se enviar a mensagem para apenas uma aplicação inscrita no canal;
    - Múltiplos valores, quando deseja-se enviar a mensagem para várias, mas não todas as

aplicações inscritas no canal.

3.5.2 JSON

O formato JSON dessa classe é conforme é apresentado abaixo:

```
{
     “channel” : <identificação do canal da mensagem>,
     “source” : <objeto MessageHeader serializado>,
     “destinations” : [<id da aplicação de destino 1>,
<id da aplicação de destino 2>, ...]
}
```

3.5.3 Erros

A desserialização dessa classe pode gerar os seguintes erros:

- MISSING_ARGUMENT: Caso “`channel`” ou “`source`” não seja informado;
- INVALID_VALUE: Caso qualquer um dos dados que compõem a mensagem não apresente um valor válido, que no caso dessa mensagem seria a identificação de canal ou aplicações que não existem;
- INVALID_FORMAT: Caso o JSON informado para a classe apresente um formato inválido.

3.6 Classe RegistrationHeader

3.6.1 Descrição

Esta classe define o cabeçalho de uma mensagem que tem por objetivo incluir ou excluir o cadastro de uma aplicação no MUIA. Ela deve conter os seguintes dados:

- **Nome da Aplicação**: Uma string obrigatória que representa o nome da aplicação cliente que irá efetuar a operação de registro no MUIA;
- **Endereço da Aplicação**: Uma string obrigatória apenas quando a ação de registro é `true`, que define o endereço IP da aplicação cliente que irá efetuar a operação de registro no MUIA;
- **Porta de Conexão da Aplicação**: Um número inteiro obrigatório apenas quando a ação é `true`, que identifica a porta na qual a aplicação cliente irá receber mensagens pelo MUIA;
- **Ação de Registro**: Um valor booleano que indica que a aplicação cliente está solicitando registro, caso seja `true`, ou que ela está solicitando a remoção do registro, caso seja `false`.

3.6.2 JSON

O formato JSON dessa classe é conforme é apresentado abaixo:

```
{
     “app-name” : <nome da aplicação>,
     “app-address” : <endereço IP da aplicação>,
     “app-port” : <porta da aplicação>,
     “register” : <ação de registro>
}
```

3.6.3 Erros

A desserialização dessa classe pode gerar os seguintes erros:

- MISSING_ARGUMENT: Caso qualquer um dos dados do cabeçalho não seja informado;
- INVALID_VALUE: Caso qualquer um dos dados que compõem a mensagem não apresente um valor válido;
- INVALID_FORMAT: Caso o JSON informado para a classe apresente um formato inválido.

3.7 Classe ChannelingHeader

3.7.1 Descrição

Como pode ser visto pela Figura 1, esta classe é abstrata, pois ela visa definir os cabeçalhos específicos para as mensagens de operações relacionadas a canais. Sendo assim, essa classe não apresenta uma serialização específica, pois essa serialização parte das suas subclasses concretas.

3.8 Classe ChannelCreatingHeader

3.8.1 Descrição

Esta classe define o cabeçalho de uma mensagem que tem por objetivo criar um novo canal para enviar mensagens entre aplicações. Ela deve conter os seguintes dados:

- **Descrição do Canal**: Uma string obrigatória que descreve brevemente o canal que será criado;
- **Quantidade Máxima de Inscritos**: Um número inteiro obrigatório que define o número máximo de aplicações clientes que podem se inscrever no canal que será criado. Caso seja informado 0, o canal não terá limite de inscritos;
- **Quantidade Máxima de Tentativas de Reenvio**: Um número inteiro obrigatório que define a quantidade máxima de vezes que o canal tentará reenviar uma mensagem;
- **Intervalo de Repetição**: Um número inteiro obrigatório que define o tempo de espera, em milissegundos, entre as tentativas de repetição de envio de uma mensagem a uma aplicação;
- **Timeout**: Um número inteiro obrigatório que define a o tempo máximo, em milissegundos, que um canal pode aguardar como resposta de uma aplicação para a qual foi enviada uma mensagem.

3.8.2 JSON

O formato JSON dessa classe é conforme é apresentado abaixo:

```
{
     “description” : <descrição do canal>,
     “max-subscribers” : <quantidade máxima de inscritos>,
     “max-retries” : <quantidade máxima de tentativas de envio>,
     “retry-interval” : <intervalo de repetição>,
     “timeout” : <timeout>
}
```

3.8.3 Erros

A desserialização dessa classe pode gerar os seguintes erros:

- MISSING_ARGUMENT: Caso qualquer um dos dados do cabeçalho não seja informado;
- INVALID_VALUE: Caso qualquer um dos dados que compõem a mensagem não apresente um valor válido;
- INVALID_FORMAT: Caso o JSON informado para a classe apresente um formato inválido.

3.9 Classe SubscribeHeader

3.9.1 Descrição

Esta classe define o cabeçalho de uma mensagem que tem por objetivo inscrever e desinscrever uma aplicação em um canal. Ela deve conter os seguintes dados:

- **Identificação do Canal**: Uma string obrigatória que identifica o canal no qual a aplicação cliente quer efetuar a operação de registro;
- **Identificação do Cliente**: Uma string obrigatória que identifica a aplicação cliente que quer efetuar a operação de registro;
- **Ação de Inscrição**: Um valor booleano que indica que a aplicação cliente está solicitando inscrição, caso seja `true`, ou que ela está solicitando a remoção da lista de inscritos, caso seja `false`.

3.9.2 JSON

O formato JSON dessa classe é conforme é apresentado abaixo:

```
{
     “channel” : <identificação do canal>,
     “client” : <identificação do cliente>,
     “subscribe” : <ação de inscrição>
}
```

3.9.3 Erros

A desserialização dessa classe pode gerar os seguintes erros:

- MISSING_ARGUMENT: Caso qualquer um dos dados do cabeçalho não seja informado;
- INVALID_VALUE: Caso qualquer um dos dados que compõem a mensagem não apresente um valor válido, que no caso dessa mensagem seria a identificação de canal ou aplicações que não existem ou ainda valores diferentes de `true` ou `false` para a ação de registro;
- INVALID_FORMAT: Caso o JSON informado para a classe apresente um formato inválido.

3.10 Classe MessageData

3.10.1 Descrição

Esta classe define como deve ser uma mensagem que será enviada entre duas aplicações cliente. Ela deve conter o seguinte dado:

- **Valor da Mensagem**: Um array de bytes que compõe a mensagem que a aplicação cliente quer enviar pelo canal. O único requisito para esse dado é que ele seja enviado serializado no formato Base64.

3.10.2 JSON

O formato JSON dessa classe é conforme é apresentado abaixo:

```
{
     “value” : <valor da mensagem>
}
```

3.10.3 Erros

A desserialização dessa classe pode gerar os seguintes erros:

- MISSING_ARGUMENT: Caso o valor da mensagem não seja informado;
- INVALID_VALUE: Caso valor não esteja no formato Base64;
- INVALID_FORMAT: Caso o JSON informado para a classe apresente um formato inválido.

4 Operações

O MUIA admite basicamente três tipos de operações básicas, já mencionadas indiretamente na seção 3, que estão listadas nas subseções dessa seção. O retorno de cada uma das operações é uma string no formato JSON, descrita com mais detalhes em cada uma das operações.

4.1 Operação de Registro

4.1.1 Descrição

Essa operação visa permitir a inclusão e exclusão de aplicações clientes na lista de aplicações que podem utilizar o MUIA.

4.1.2 Entrada

Para executar tal operação, a aplicação cliente deve enviar um pacote com um cabeçalho de mensagem do tipo `RegistrationHeader`, descrito na seção 3.6. Além disso, pra essa operação, não é necessário enviar uma instância de `MessageData`.

4.1.3 Retorno

O retorno do MUIA para essa operação é composto dos seguintes dados:

- `status`: Resultado da operação, podendo assumir os seguintes valores:

CÓDIGO	SIGNIFICADO
10	Operação realizada com sucesso.
11	Erro indicando que aplicação não foi registrada pois já existe uma aplicação com o mesmo nome registrada no MUIA.
12	Erro indicando que a aplicação não foi removida da lista de registro pois não existe uma aplicação com aquela identificação.
13	Erro indicando que algum erro desconhecido aconteceu no momento da aplicação da ação de registro.

O retorno, no formato JSON, fica da seguinte forma:

```
{
     “status” : <status da operação>
}
```

4.2 Operação de Canal

Essa operação lida com canais e é dividida em duas operações específicas:

4.2.1 Operação de Criação de Canal

4.2.1.1 Descrição

Essa operação visa permitir a criação de canais definidos pelas próprias aplicações clientes para que as aplicações possam utilizá-lo na troca de mensagens.

4.2.1.2 Entrada

Para realizar essa operação, a aplicação cliente deve enviar um pacote com cabeçalho de mensagem do tipo `ChannelCreatingHeader`, definido na seção 3.8. Além disso, pra essa operação, não é necessário enviar uma instância de `MessageData`.

4.2.1.3 Retorno

O retorno do MUIA para essa operação é composto dos seguintes dados:

- `status`: Resultado da operação, podendo assumir os seguintes valores:

CÓDIGO	SIGNIFICADO
20	Operação realizada com sucesso.

21	Erro indicando que não foi possível criar o canal.

- `channel-id`: Identificação do canal criado. Essa chave deverá ser utilizada pela aplicação para enviar mensagens para o canal.

O retorno, no formato JSON, fica da seguinte forma:

```
{
     “status” : <status da operação>,
     “channel-id” : <identificação do canal criado>
}
```

4.2.2 Operação de Inscrição em Canal

4.2.2.1 Descrição

Essa operação visa permitir a inclusão e exclusão de aplicações clientes na lista de aplicações inscritas em um canal.

4.2.2.2 Entrada

Para realizar essa operação, a aplicação cliente deve enviar um pacote com cabeçalho de mensagem do tipo `SubscribeHeader`, definido na seção 3.8. Além disso, pra essa operação, não é necessário enviar uma instância de `MessageData`.

4.2.2.3 Retorno

O retorno do MUIA para essa operação é composto do seguinte dado:

- `status`: Resultado da operação, podendo assumir os seguintes valores:

CÓDIGO	SIGNIFICADO
30	Operação realizada com sucesso.
31	Erro indicando não foi possível inscrever a aplicação no canal.
32	Erro indicando não foi possível remover a aplicação da lista de inscritos no canal.

O retorno, no formato JSON, fica da seguinte forma:

```
{
     “status” : <status da operação>
}
```

4.3 Operação de Mensagem

4.3.1 Descrição

Esta operação visa permitir o envio de mensagens entre aplicações, tanto entre MUIA e Cliente quanto MUIA e MUIA.

4.3.2 Entrada

Para executar essa operação, a aplicação remetente deve enviar um pacote com cabeçalho de mensagem do tipo `MessagingHeader` , descrito na seção 3.5, e ainda é obrigada a enviar uma

instância de `MessageData`, descrita na seção 3.10, pois essa instância vai conter a mensagem que de fato será enviada para a aplicação destinatária final.

4.3.3 Retorno

O retorno do MUIA para essa operação é composto do seguinte dado:

- `status`: Resultado da operação, podendo assumir os seguintes valores:

CÓDIGO	SIGNIFICADO
40	Operação realizada com sucesso.
41	Erro indicando não foi possível receber a mensagem no canal.

O retorno, no formato JSON, fica da seguinte forma:

```
{
     “status” : <status da operação>
}
```

5 Códigos de Erros

Os códigos de erro definidos na seção 3 em cada uma das descrições de mensagem são listados abaixo, bem como seus valores numéricos e uma descrição geral sobre o erro, que são os valores retornados para as aplicações que enviam mensagens para o MUIA. Além disso, outros códigos de erros que podem acontecer em todas as mensagens são descritos abaixo.

CÓDIGO	ERRO	SIGNIFICADO
1	MISSING_ARGUMENT	Algum argumento obrigatório para a mensagem está ausente.
2	INVALID_VALUE	Algum argumento para a mensagem está com um valor inválido.
3	INVALID_FORMAT	A entrada para a mensagem está com formato inválido.
9	UNKNOWN	Algum erro desconhecido ocorreu.